IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

FRANCIS, H. et al.

Atty. Ref.: 550-450; Confirmation No. 1418

Appl. No. 10/648,293

TC/A.U. 2183

Filed: August 27, 2003

Examiner: Pan, Daniel H.

For: EXECUTING VARIABLE LENGTH INSTRUCTIONS STORED WITHIN A PLURALITY OF DISCRETE MEMORY ADDRESS REGIONS

* * * * * * * * * * *

October 29, 2007

Mail Stop AF
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ARGUMENTS IN SUPPORT OF PRE-APPEAL BRIEF REQUEST FOR REVIEW

I. Claims 43-62 Recite Statutory Subject Matter[1]

Claim 43 recites: "A computer program product including a storage medium readable by a data processing apparatus encoded with instruction code which, when executed by the data processing apparatus, controls the data processing apparatus to execute a sequence of variable length instructions stored within a plurality of discrete memory address regions within a memory of said data processing apparatus." The Examiner attempts to substitute the language from page 15, lines 6-13 in the specification for what is claimed—a clear error. A proper analysis under §101 is based on what is claimed. The quoted portion of claim 43 makes clear that the claim does not recite "pure software" because the product includes instruction code embodied in a storage medium readable which is executable by a data processing apparatus to achieve useful, concrete, and tangible results, as explained in the response filed Nov. 21, 2006, pages 12-14.

[1] Numbered paragraph 5 in the final action does not make sense in light paragraph 2.

Moreover, a person of ordinary skill in the art would have understood that statements on page 15, lines 6-13 in the specification were made with the understanding that software-based control relates to a software program controlling a computer to perform an operation and that hardware control relates to hardware performing an operation like fetching, decoding, and executing an instruction. See page 10, lines 21-24 of the specification: "At step 28, a single step flag is set within a register of a Javacard decoder controlling coprocessor CP14. Such control registers are generally conveniently accessible under software control (e.g. coprocessor register store instructions) and can be used by programs to pass configuration information to hardware." The registers, decoder, and coprocessor are hardware, and software control relates to instructions that control a processor.

The Examiner's position is out of touch with the reality of technology today in which computer program products form the foundation of so many modern day innovations. How else can an inventor protect against others selling commercial computer program products that utilize the inventor's innovations? Inventors have a right to stop such computer product infringers as well as infringers who actually use a computer controlled by that product. Indeed, searching for the term "computer program product" on the U.S. patent office's patent database returns over 20,000 granted U.S. patents that use that language. The undersigned reviewed the claims of many of these patents and found issued claims using much the same format as used in claim 43.

II. The Claimed Technology and Ohshima

This application relates to executing a variable length instruction stored in distinct memory locations. The first part of a variable length instruction is stored in a first memory location, and the second part is stored in a second different memory location. A "fix-up" memory address region is used to bring together the two parts of the single instruction. When a

two-part, variable-length instruction occurs, the program execution flow is temporarily diverted to the fix-up memory to read the freshly reconstructed variable length instruction stored there. Thereafter, the program execution flow is returned to reading from the normal memory.

Ohshima describes an instruction that includes both basic and expanded segments. A basic segment contains a code indicating the type of instruction (basic or expanded), and an expanded segment contains information relevant to the type of instruction specified by the basic segment. One instruction is formed from one or more basic and expanded segments. The length of expanded segments can vary, and this length is not known until its corresponding basic segment has been decoded. Hence, in a situation where one of Ohshima's instructions consists of two basic segments and the first basic segment is followed by an expanded segment, the second basic segment cannot be input into a decoder until after the first basic segment has been decoded, which allows the length of the expanded segment to be determined. Thus, two cycles are required to decode the single instruction. See col. 2, line 59 to col. 3, line 9.

III. Ohshima and Watt Fail To Teach The Concatenation Into Fix-up Memory

Ohshima does not teach concatenating instruction data from two distinct memory address regions "into a fix-up memory address region of said memory to form concatenated instruction data containing said variable length instruction" as recited in claim 1. If there is a concatenation of instruction data in Ohshima, it is performed at the instruction code bus 14 which receives the rearranged segments from block 8 (see Figs. 6A and 6B). Applicants respectfully submit that a person skilled in the data processing art would not reasonably equate Ohshima's instruction code bus to the claimed fix-up memory region—a bus and a memory are two different things with two different functions and operations.

The Examiner refers to a "rearranged pointer P" in Figs. 4 and 8C. Such a pointer is not found in either of these figures. (Fig.1 shows a pointer P1 that points to the instruction code string C before it is rearranged on the instruction code bus.) Nor would a person skilled in the data processing art equate an address pointer with a fix-up memory region. A pointer can point to a memory address, but a pointer is not memory. The Examiner further refers to Figs. 2a-2b. They should be understood in light of Fig. 1 which shows that the instruction code string segments are re-arranged onto the instruction code bus "by preliminarily separating the bus into the base segment fields ... and the expanded segment fields." Col. 5, lines 60-67 (emphasis added). In Fig. 2, the rearranged segments are provided from the instruction code bus to the instruction decoder 9 and instruction register 10. Fig. 2 "shows some examples of the arrangements of the expanded segments" where "the basic segment is connected to the upper side X of the bus and the expanded section outputs to the correspond expanded segment field Y [of the bus]." Col. 6, lines 10-14 (emphasis added). Any concatenating occurs on the instruction code bus and not in a fix-up memory.

IV. Ohshima and Watt Fail To Teach Diverting Program Execution to Fix-up Memory And Then Restoring Program Execution

The Examiner maps the claimed diverting and restoring to execution of an instruction on Ohshima's instruction code bus and execution of the next instruction. But the instruction code bus is not part of the memory address space. Also, instruction decoding always takes place from Ohshima's instruction code bus (see Fig.1). So there is no need to divert program execution flow to the instruction code bus or restore program execution flow from the instruction code bus.

The Examiner refers to a rearranged pointer P in Figs. 4 and 8C, which again is not labeled in those Figures. The reader pointer 407 in Fig. 4 simply instructs which segments will be routed to specific instruction code bus lines 414. Read out pointer 7 in Figure 8B does the

same thing via output position identification circuit 20, with the rearranged segments being output onto the instruction code bus 14_1 and 14_2. Program execution flow is not being diverted to another portion of memory to execute the rearranged instruction. As shown in Fig. 1, instruction execution always proceeds from the instruction code bus to the decoder 9 and instruction register 10.

Ohshima lacks the claimed restoring operation. Because there is no diversion to fix-up memory in Ohshima, there is no need for restoration of program execution flow from the fix-up memory. The Examiner refers to Fig. 5. But this figure further confirms that the rearranged instruction is provided to the instruction code bus before execution. See "arrangement on instruction code bus 520" at the bottom of Fig. 5.

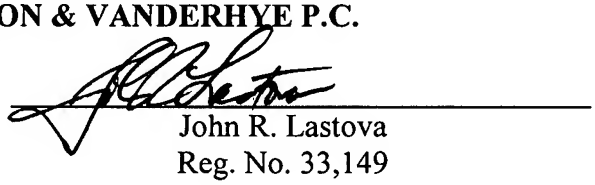
V. The Rationale To Modify Ohshima Is Insufficient

The Examiner speculates that Watt's abort signal "could be implemented" in Ohshima to avoid "data loss due to improper memory address space access." But the question is why do that in Ohshima? There is no concern of improper memory access in Ohshima. And if such an abort was used in Ohshima, it would likely come when the instruction code C is retrieved--not during the subsequent rearrangement of segments.

The final rejection should be withdrawn and the application passed to allowance.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: 

John R. Lastova
Reg. No. 33,149

901 North Glebe Road, 11th Floor
Arlington, VA 22203-1808
Telephone: (703) 816-4000
Facsimile: (703) 816-4100